(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 February 2002 (07.02.2002)

PCT

(10) International Publication Number
WO 02/11357 A2

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US01/21428

(22) International Filing Date: 6 July 2001 (06.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/627,623 28 July 2000 (28.07.2000) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, M/S: UPAL01-521, Palo Alto, CA 94303 (US).

(72) Inventors: UHLER, Stephen; 330 Mundell Way, Los Altos, CA 94022 (US). DIGIORGIO, Rinaldo; 20 Mile Common Road, Easton, CT 06612 (US).

(74) Agents: HECKER, Gary, A. et al.; The Hecker Law Group, 1925 Century Park East, Suite 2300, Los Angeles, CA 90067 (US).

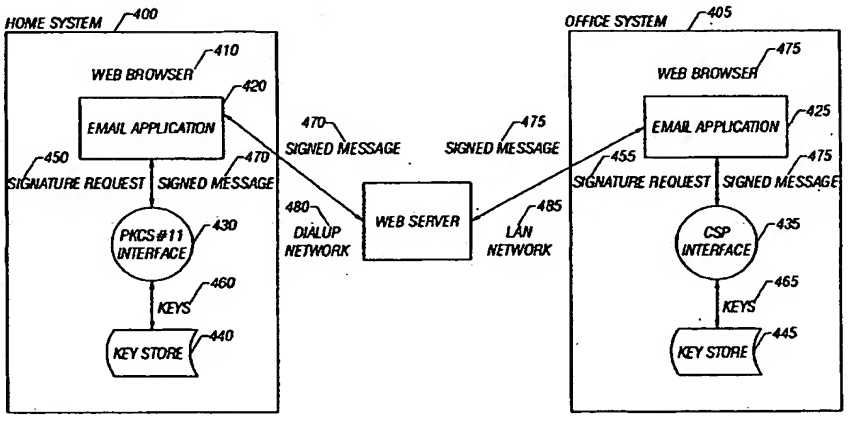
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR CRYPTOGRAPHIC KEY MANAGEMENT USING URL PROGRAMMING INTERFACE



(57) Abstract: A method and apparatus for managing cryptographic keys using URL (Uniform Resource Locator) programming interface (UPI) is described. A user of public key, symmetric, or other cryptographic scheme must maintain one or more keys, a list of trusted certificate authorities, and a database of certificates. This collection of items is called a *data vault*. Applications, including Web browsers, use different standards, algorithms, and formats to store and manage cryptographic keys. The claimed invention provides a single data vault that is accessible by any HTTP-capable application. The data vault may reside on any form of storage including a disk file, a directory, a Smart Card, a desktop computer, a handheld device, or other similar devices. A data vault may also extend over numerous combinations of storage devices. This architecture, in one embodiment of the invention, provides a data vault as a very small HTTP stack. This minimal stack may be installed on a variety of devices to permit secure Internet access to and from any HTTP-capable device.

WO 02/11357 A2

METHOD AND APPARATUS FOR CRYPTOGRAPHIC KEY MANAGEMENT USING URL PROGRAMMING INTERFACE

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

This invention relates to the field of computer systems, and, more specifically, managing cryptographic keys using URL (Uniform Resource Locator) programming interface (UPI).

BACKGROUND ART

Secure network access to files, processes and devices poses many challenges to application programmers. A lack of standards creates a multitude of interfaces to security modules. Many World Wide Web (Web) browser

programs use different cryptographic systems and algorithms, providing complex interface requirements for Web applications.

Users of network security in the current technology face challenges as well. A user with multiple network access devices will invariably have a plethora of security data to manage across these devices. Access devices, for example, personal computers, hand-held computers, laptop computers, wireless Web devices and networked office systems, will often use different cryptographic schemes to identify a user to the network.

Unfortunately, cryptographic data is difficult to transfer between access devices. If the user wishes to send or receive encrypted data or messages the user will likely maintain cryptographic "key" data on each of these devices.

Additionally, a user of digital signatures must maintain one or more sets of digital certificates from a digital certificate authority that authenticates the user's digital signature. Data such as cryptographic keys, trusted certificates and a database of digital certificates stored in one physical location will be referred to herein as cryptographic data, or as a *data vault*. A data vault resides on one or more portable storage mediums and contains distinct elements of cryptographic data. Each cryptographic element of the data vault is stored separately and is individually addressable.

Secure Network Applications

Using current technology, writing application programs requiring secure access to devices such as home automation, home networks, or realtime process control devices, is also a challenge. Every device has its own communication

protocol and security considerations, which an application programmer must understand and work with in order to provide a control interface to the device. This lack of interface standards makes programming applications to provide access to these devices prohibitive. Yet, there is a growing interest in connecting devices to the Internet. While Sun Microsystems, Inc.'s Jini™ Community Process is beginning to address this issue, the lack of well-defined interfaces for well-known services is hindering widespread adoption of the Jini™ technology. Still, security is a major consideration for programmers of all Internet Aware Devices (IADs). While the benefits of direct connection of household and other devices are revolutionary, the opportunity for unauthorized access greatly increases.

Many of the current approaches to solving this problem use proprietary solutions that do not scale or make effective use of the Internet. A workable solution must provide strong authentication, offer optional encryption of the established session, and operate without requiring special permission to reconfigure firewalls. Additionally, the solution should not make use of large applets that only work on corporate LANs, as some users may access the system from low-speed modem and wireless connections.

Applications for Secure Network Access

Many software applications suffer from a lack of secure device interfaces. Development of applications to access IADs is costly and software life cycles grow shorter as technology goes through rapid changes. Current architectures do not meet the needs of software developers to solve the problems of IAD access for desirable applications such as home networks, home automation, and process control, to name but a few. While many other applications are

susceptible to the same problems, the example of home networks is used here to illustrate the problems of using current secure software architectural models.

Home Networks

The number of users with digital access devices is growing. However, a single user may have many such devices spread out over two or more locations such as home, office, and car. Home networks now provide access to share peripherals, computers, and other devices like environmental controls and home entertainment systems. Access to a home network is valuable to users away from their homes, just as it is to those away from their offices. Many home systems contain important data a user many need to access from the office. For example, leaving that key document on your home system after a long weekend of work can be a disaster when you reach the office without it. Similarly, during a hectic business day a user may need access to personal data stored in their home computer.

Traditional access architectures do not provide the sort of security features required to safely and securely access and manage a home network remotely. An architecture is needed that can provide controlled access for tailored custom services. The current undesirable alternative is to teach all users of a home network how to use traditional access mechanisms such as FTP and telnet. However, these programs are difficult for non-technical users to master and fraught with possible damaging errors. Additionally, secure versions of these services are not always available on all portable access devices. What is needed is a more pragmatic approach - - allowing current devices to access the home network through a simple interface, but only after users have given the appropriate cryptographic credentials.

Problems with Web Applications and IADs

Web based applications need easy access to devices that require a secure interface, as discussed above. Unfortunately, current technology requires Web applications to utilize Web browsers to provide a secure interface for communication between a user and a Web-based device. However, the primary browser technologies available today, Netscape and Internet Explorer, use different cryptographic interfaces. Netscape utilizes a scheme called PKCS (RSA Labs' Public Key Cryptographic Standard) and Internet Explorer uses a scheme called CSP (Microsoft's Cryptographic Service Provider.) These two systems have different Application Program Interfaces (APIs), though their basic functions are similar. This dichotomy requires Web applications to manage their cryptograph interface accordingly.

Other available browsers may add additional complication for Web applications developers that want to provide secure access to IADs. Various browsers may use Public Key encryption, using public and private keys, or possibly use a symmetric encryption scheme that uses a session key. In addition, encryption schemes are implemented using different algorithms, provoking even further complication for applications relying on the Web browser to provide a secure interface to an IAD. Cipher modes, salt values, and other factors affect the ability of a Web application to interface to any given browser's secure interface.

Storing Cryptographic Keys

Complications exist for users of Web applications as well. In the present technology, Web browsers are responsible for generating and storing keys,

using the various security modules available to them. Therefore, to use multiple browsers, for example Netscape and Internet Explorer, users discover they must keep multiple cryptographic keys. When users keep keys at home, at work, and on portable devices key management becomes burdensome and security decreases. Furthermore, keys kept on networked hard disks are subject to various forms of hacker attacks.

The present technology faces many disadvantages, including:

1. Different security models in Internet Explorer and Netscape
2. No uniform interface to certificate access methods
3. Redundant storage when both systems are used
4. Portability of certificates is difficult
5. Certificate management requires two different interfaces
6. Other applications cannot easily use certificates
7. Integration with Smart Cards has similar competing models problems

In the Prior Art, each cryptographic scheme has a unique interface leading to a lack of portability. Having duplicate policies for certificates in multiple applications complicates the task and decreases the security -- thereby providing multiple points of entry that could lead to the private key is that certificates contained.

ADDITIONAL BACKGROUND

To better understand the problem solved by the claimed invention, a brief description of some prior art technologies follow.

General Background Material About Computer Networks

In order to facilitate an understanding of how computer networks allows for the transfer of data a brief discussion about such networks follows.

Computers and computer networks are used to exchange information in many fields such as media, commerce, and telecommunications, for example. The exchange of information between computers typically occurs between a "server application" that provides information or services, and a "client application" or device that receives the provided information and services. Multiple server applications are sometimes available on a "system server" such as a single computer server that provides services for multiple clients. Alternatively, distributed server systems allow a single client to obtain services from applications residing on multiple servers. For example, in current distributed server systems, client applications are able to communicate with server applications executing on the same computer system or on another computer system accessible via a network, for instance via the Internet.

The Internet is a worldwide network of interconnected computers. An Internet client computer accesses a computer on the network via an Internet provider. An Internet provider is an organization that provides a client (computer) with access to the Internet (via analog telephone line or Integrated Services Digital Network line, for example). A client can, for example, read information from, download a file from, or send an electronic mail message to another computer/client using the Internet.

To retrieve a file or service on the Internet, a client must typically search for the file or service, make a connection to the computer on which the file or service is stored, and download the file or access the service. Each of these steps

may involve a separate application and access to multiple, dissimilar computer systems (e.g., computer systems having operating different systems). The World Wide Web (Web) was developed to provide a simpler, more uniform means for accessing information on the Internet.

The components of the Web include browser software, network links, servers, and Web protocols. The browser software, or browser, is a tool for displaying a user-friendly interface (i.e., front-end) that simplifies user access to content (information and services) on the Web. Browsers use standard Web protocols to access content on remote computers running Web server processes. A browser allows a user to communicate a request to a Web server without having to use the more obscure addressing scheme of the underlying Internet. A browser typically provides a graphical user interface (GUI) for displaying information and receiving input. Examples of browsers currently available include Netscape Navigator and Communicator, and Microsoft Internet Explorer.

Web browsers and servers communicate over network links using standardized messages formats called protocols. The most common modern protocol is the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite. The protocols are based on the OSI (Open Systems Interconnect) seven-layered network communication model. Web messages are primarily encoded using Hypertext Transport Protocol (HTTP). HTTP instantiates the (top) Application layer of the OSI model. Application layer protocols facilitate remote access and resource sharing and are supported by the reliable communications ensured by the lower layers of the communications model. Therefore, HTTP simplifies remote access and resource sharing between clients and servers while providing reliable messaging on the Web.

Information servers maintain the information on the Web and are capable of processing client requests. HTTP has communication methods that allow clients to request data from a server and send information to the server.

To submit a request, the client browser contacts the HTTP server and transmits the request to the HTTP server. The request contains the communication method requested for the transaction (e.g., GET an object from the server or POST data to an object on the server). The HTTP server responds to the client by sending a status of the request and the requested information. The connection is then terminated between the client and the HTTP server.

A client request, therefore, consists of establishing a connection between the client and the HTTP server, performing the request, and terminating the connection. The HTTP server typically does not retain any information about the request after the connection has been terminated. That is, a client can make several requests of an HTTP server, but each individual request is treated independent of any other request.

The Web employs an addressing scheme is that uniquely identifies Internet resources (e.g., HTTP server, file, or program) to clients and servers. This addressing scheme is called the Uniform Resource Locator (URL). A URL represents the Internet address of a resource on the Web. The URL contains information about the protocol, Internet domain name and addressing port of the site on which the server is running. It also identifies the location of the resource in the file structure of the server.

HTTP provides a mechanism of associating a URL address with active text. A browser generally displays active text as underlined and color-coded.

When activated (by a mouse click, for example) the active text causes the browser to send a client request for a resource to the server indicated in the text's associated URL address. This mechanism is called a hyperlink. Hyperlinks provides the ability to create links within a document to move directly to other information. A hyperlink can request information stored on the current server or information from a remote server.

If the client requests a file, the HTTP server locates the file and sends it to the client. An HTTP server also has the ability to delegate work to gateway programs. The Common Gateway Interface (CGI) specification defines a mechanism by which HTTP servers communicate with gateway programs. A gateway program is referenced using a URL. The HTTP server activates the program specified in the URL and uses CGI mechanisms to pass program data sent by the client to the gateway program. Data is passed from the server to the gateway program via command-line arguments, standard input, or environment variables. The gateway program processes the data and returns its response to the server using CGI (via standard output, for example). The server forwards the data to the client using the HTTP.

When a browser displays information to a user it is typically as pages or documents (referred to as "web pages"). The document encoding language used to define the format for display of a Web page is called Hypertext Markup Language (HTML). A sever sends a Web page to a client in HTML format. The browser program interprets the HTML and displays the Web page in a format based on the control tag information in the HTML.

Current network systems provide a way to transfer and display data. However, these network systems have left the management of cryptographic data to web browsers, application developers, and users to manage without standards or coordinated storage methods. The prior art therefore lacks a method for managing cryptographic keys in a modern network environment.

SUMMARY OF THE INVENTION

A method and apparatus for managing cryptographic keys using URL (Uniform Resource Locator) programming interface (UPI) is described. A user of public key, symmetric, or other cryptographic scheme must maintain one or more keys, a list of trusted certificate authorities, and a database of certificates. This collection of distinct items is called a *data vault*. Current methods of storing such information are generally incompatible with each other. Applications, including Web browsers, use different standards, algorithms, and formats to store and manage cryptographic keys.

The claimed invention provides a single data vault that is accessible by any HTTP-capable application. The data vault may reside on any form of storage including a disk file, a directory, a Smart Card, a desktop computer, a handheld device, or other similar devices. A data vault may also extend over numerous combinations of storage devices. This architecture uses, in one embodiment of the invention, the Java™ language to provide a technology that allows users to deploy the data vault interface as a very small protocol (HTTP) stack. In one embodiment of the invention, the protocol stack has a core of less than 100 KB. However, the invention encompasses protocol stacks of both larger and smaller size. A minimal stack may be installed on a variety of devices to permit secure Internet access to and from any HTTP-capable device.

One kind of data managed by the data vault includes digital certificates. Digital Certificates are generally issued by a Digital Certificate Authority (DCA). However, there is nothing to prevent the claimed invention from issuing its own digital certificates.

Single source management of cryptographic data makes connecting legacy architectures, home networks, corporate LANs, and Internet-aware devices (IADs) to Internet interface units (IIUs) manageable and secure.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow diagram of the overall process in one embodiment of the claimed invention.

Figure 2 is a representation of the functions provided by the claimed invention.

Figure 3 represents a smart card reader, in one embodiment of the claimed invention.

Figure 4 is a representation of the current technology's approach using multiple key stores to generate digital signatures.

Figure 5 shows the method of one embodiment of the claimed invention for generating digital signatures.

Figure 6 shows, in one embodiment of the claimed invention, a session that creates a key pair and signs a data item using the claimed invention.

Figure 7 is a block diagram of one embodiment of a computer system capable of providing a suitable execution environment for an embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention is a method and apparatus for managing cryptographic keys using Uniform Resource Locator (URL) programming interface (UPI). In the following description, numerous specific details are set forth to provide a more thorough description of embodiments of the invention. It will be apparent, however, to one skilled in the art, that the invention may be practiced without these specific details. In other instances, well known features have not been described in detail so as not to obscure the invention.

Though discussed herein with respect to the Java™ programming language, the invention may be implemented in any environment that supports object or data access through references, and that provides information about private key URL programming interfaces. Additionally, while one embodiment of the present invention is described using Java Card™ technology, the invention encompasses the use of any device that provides portable storage of programmable data.

URL Programming Interface (UPI)

Developing consistent, reliable Web applications that interface to different devices--such as home networks, home automation systems, or real-time process control devices -- can be vastly simplified by treating devices as URLs. The URL programming interface (UPI) effectively provides a set of URLs for a device that is available to any application capable of performing the HTTP (HyperText Transfer Protocol.) HTTP has become so universal that in college computer courses students are often given as an assignment the creation of an HTTP stack. This trend is due to the growing interest in connecting devices to

the Internet. Sun Microsystems Laboratories has used the Java™ language to develop a technology that allows users to deploy a very small HTTP stacks. However, the claimed invention can be practiced using any computer programming language and transmission protocol.

These small HTTP servers can be run on any embedded device or used as application servers that are similar to traditional Web servers. Such minimal servers provide an integrated presentation and service layer for a device. Because a Web server answers URL requests, universal access to devices from any Internet node is achieved. UPI provides an interface to allow applications, including Web applications, to access any device as if the device is a file resident on a Web server, if the device can perform the HTTP protocol.

The UPI architecture, at its core, consists of a series of handlers that are similar to servlets, but have fewer features. These handlers are mapped to incoming URL requests. This allows the UPI function on very small devices, such as a TINI board. The UPI supplies handlers for Web services such as file service and CGI script execution. Application developers can develop handlers for Internet aware devices (IADs) simply by coding a few required methods, such as *init* and *respond*, along with code that understands the applicable device grammar.

The Solution to Multiple Data Vaults

The claimed invention provides a solution for users required to maintain multiple data vaults for various applications and network access devices. All cryptographic data using the claimed invention may be stored on a single Portable Secure Identification Device (PSID.) In one embodiment of the invention, the PSID may be implemented using a Smart Card. However, the

invention may be practiced using other physical manifestations of the PSID, if such devices provide sufficient data storage and are, or can be made to be, Internet Aware Devices (IADs.) A Personal Data Assistant (PDA) is another example of a PSID that may be an IAD, if it is equipped with a modem or wireless Web connection.

Smart Cards

A Smart Card is a credit card-sized plastic card with an integrated circuit (IC) inside. The IC contains a microprocessor and memory, which gives Smart Cards the ability to process, as well as store more information than was previously possible.

In the case of a Java Card™ platform-based card, Java technology-based applications, in the form of byte-codes, are loaded into the memory zone of the Smart Card's microprocessor where they are run by the Java Virtual Machine. The executable code is platform independent so that any card incorporating a Java Card™ interpreter can run the same application.

Multiple Java™ technology-based applications can reside on a single card, each allocated to their own secure memory areas to ensure their integrity and eliminate program tampering, either by individuals or through program interference.

Problems with Smart Cards

Currently, using Smart Cards on various systems is still subject to errors. It is expensive to deploy due to conversion issues and competing browser security models. There is currently a lack of standards and no clear business model to inspire development. Additionally, there is little interoperability

between card-based applications developed by various service providers such as telecommunications operators, retail vendors, and financial institutions.

The current concept of a Web browser owning both the Smart Card and the API interface to the card decreases the utility of a card that only works with a browser from company A or company B. There still many applications that need to have access to Smart Cards independent of the browser. It would be better to have the resource -- a Smart Card, in this case -- appear as a shared resource that is available to all applications on the network, both local and remote.

Solving the Smart Card Problem with UPI

In one embodiment, the claimed invention solves this problem by creating an architecture utilizes the latest versions of Java™ and Java Card™ and that is independent of Microsoft and Netscape. The architecture also supports non-Java™ Smart Cards. This architecture is built on the concept of UPIs that create small Web servers, which in turn are wrapped around physical or logical devices. Java Cards™, in one embodiment of the invention, allow applications to easily and reliably verify the authority of a user to perform UPI operations.

As mentioned above, the best current Smart Card solutions on the market combines with plethora of methods for accessing the Internet to pose a challenge for MIS and IT department. The problems are even more acute in the consumer world. The claimed invention's architecture solves this problem by making the client browser independent; with the addition of the claimed invention, UPI has the potential to make Web servers appeared to be equal in terms of authenticated access and services.

The core concept of UPI involves speaking to a Smart Card and card reader combined with an HTTP stack; in other words, the card reader and card answer to URLs to provide responses to queries as illustrated in Figure 1 and described above. Typically, responses to a Web server request are Web pages, disk files, or data generated from programs. In one embodiment of the invention, data returned to the user comes from the data vault on a Smart Card in a Smart Card reader.

Figure 1 illustrates a flow diagram of the overall process of one embodiment of the claimed invention. At step 100, a user on a Web accessible device invokes a Web browser, or other application capable of generating URL requests. At step 110, the user starts an application program that requires secure communication. For example, the user may wish to send an email message containing a digital signature by which to identify the sender. This scenario will be used throughout this description as an example of the kind of transaction the claimed invention will facilitate. However, the claimed invention may be practiced with any application that requires secure communication and/or authentication of the user over a network or other intercommunication fabric. At step 120 the user invokes a secure transaction, for example commanding the email message be sent "signed" with a digital signature. The application program calls the interface in step 130 by issuing a URL to the invention's UPI to initiate a session with the key store that contains the user's cryptographic data. The user's key store may be on any IAD that is accessible via UPI at the time the user makes the request for the data. At step 140, the application or the user selects the correct token (IAD and key store) and at step 150, a digital signature is produced. In one embodiment of the invention, the IAD is a Smart Card in a Smart Card reader equipped with the present invention's UPI. Finally, at step 160 the signed data is returned to the application for further processing.

Figure 2 represents the functions provided by one embodiment of the claimed invention. Functions 200 and 205 provide the ability to login or logout a user with an established communication session, using the user's PIN (Personal Identification Number). A PIN is required in addition to access to the user's Smart Card to ensure that the user is authorized to access the card. Functions 210 and 215 are session management functions. A session establishes communication with a particular token, in this example a Smart Card. Functions 220 and 225 provide an application with a list of available tokens and their attributes. A token may be available or not available; for example, a Smart Card removed from a reader may be an unavailable token. The claimed invention provides the application the ability to manage objects using functions 230, 235, and 240. Objects allow an application to access and manage information retrieved from or stored to a token. Functions 245 and 250 control an object's attributes. Attributes are label value pairs of information retrieved from or stored to a token. Finally, functions 255 and 260 provide decrypt and sign functions for using digital signatures. A more detailed explanation of each of these functions in one embodiment of the invention follows.

Elements of the UPI

As an illustration of one embodiment of the invention, handlers for the PKCS#11 system used by Netscape are defined as follows. The invention may be practiced with or without the following functions, which are provided herein only as an example of the claimed invention. The invention may be practiced using PKCS, any version of PKCS, CSP, or any other cryptographic scheme.

getTokenList – this function retrieves a list of tokenIds for all available tokens. A tokenId is assigned so that each token available to a given application at a given time has a unique tokenId. When a token is no longer available, perhaps a card removed from a reader, then the tokenId is removed from the Token List.

getTokenInfo – this function returns a list of attribute-value pairs describing the token associated with tokenId.

createSession – this function creates a session to be used to communicate with the specified token and returns a sessionId for that session (unique among all open sessions for the calling application at a given time.) Only one operation can be active for a given session at a given time.

closeSession – this function closes the session associated with the sessionId.

login – this function connects the user to the session associated with sessionId, using a PIN. All active sessions associated between this application and this token will automatically be logged in (assuming the PIN is correct.)

logout – this function ends the user login with the session identified by sessionId.

createObject – this function creates an object on the token identified, with the attributes specified. It returns an objectId associated with the new object.

deleteObject – this function deletes an object created with createObject above.

getAttributes – this function gets the value of the attribute whose identifiers are listed in the attribute from the object associated with objectId from the token.

setAttributes – this function sets the attributes listed in attrs for the object associated with objectId on the token.

findObjects – this function finds all objects on the token that match the attributes listed in attrs. An array of their objectIds is returned.

generateKeyPair – this function generates a new key pair on the token for the mechanismId which indicates the cryptographic scheme to use. The public key, when appropriate, is placed in an object with the attributes specified by publicKeyAttrs. The private key is placed in an object with the attributes privateKeyAttrs. An array of the two keys is returned.

decrypt – this function decrypts the encrypted text passed to it using the mechanism identified by mechanismId and the private key stored in the object associated with the keyId. Plain text is returned.

sign – this function signs the data using the mechanismId indicated and the private key stored in the object associated with keyId. The function returns the signature.

Smart Card Reader Example

An example of a UPI for a Smart Card reader is Provided. The client software is called *PersonalCard Services,* and the server side products are called *PersonalCard Service Providers.* Service Providers provide server implementations of code but complete client requests and/or transactions that

require server services. PersonalCard Server is a minimal HTTP 1.1 server that can respond to URLs of the form:

http://somehost:someport/SecureTokenServices/GetId.

This is an example of a UPI URL that queries a handler for the operation of the *getid* method. The last element in this path is not treated as a file or CGI script, but rather as an identifier that allows the handler secured token services to dispatch appropriate processing. The *getid* path component returns the ID of Smart Card in the reader. This approach allows non-Java programmers to develop Smart Card applications using Java™ script. This approach minimizes the IT deployment costs because it allows Smart Card readers and Smart Cards to be treated as Web servers. The servers can be maintained over the Web using HTML and JavaScript macros, or using Java programs and applications.

Further illustrating the claimed invention, Figure 3 represents a Smart Card reader, in one embodiment of the invention. The claimed invention provides a small HTTP stack that may be run on any IAD that can support a Web browser. This stack, along with the UPI interface, provides the device the ability to respond to URLs. Figure 3 shows Smart Card reader 300, in one embodiment of the invention. URL 310 enters the system via a network interface, where it is processed by local Web server 320 — a miniature HTTP stack program. This interface communicates with traditional Smart Card reader interface 340 to store and read data from Smart Card 350 and its Integrated Circuit chip 360.

Managing User Digital Certificates

Applications wish to perform many operations using a user's digital certificates. Some may include: SSL (Secure Socket Layer) protocol, Signed and

Encrypted Email, Form Signing, Single Signon, and Object Signing. Furthermore, management of certificates numerous functions must be implemented by the application. Certificate management functions include: Issuance of Certificates, Certificate and LDAP Directory management, Key Management, Renewing and Revoking Certificates, and Registration Authorities.

The claimed invention provides access to these certificate operations and certificate management functions through a series of URLs that allow application programs and/or applets to securely interact with certificates stored in the invention's data vault. The data vault can be protected with various levels of security. The data vault may reside in any location, including a local or remote data file, or on an IAD such as a Personal Digital Assistant (PDA) or a Smart Card. Some examples of URLs that provide access to a secure portable data vault are:

http://localhost:9000/GetCertificate?name="My Email Signer"?verify="signature"

http://localhost:9000/DeleteCertificate? name="My Email Signer"?verify="signature"

http://localhost:9000/NewCertificate? name="My Email Signer"?verify="signature"

http://localhost:9000/TestCertificate? name="My Email Signer"?verify="signature"

In current technology, each device manufacturer has a different interface, leading to a lack of software portability. The UPI-based approach provides centralized storage and management with stronger security. Having duplicate policies for certificates in multiple applications and multiple locations is undesirable because it complicates the task of managing certificates and

decreases security by duplication; thereby providing multiple points of entry that could lead to discovery of the private keys the certificates contain.

The claimed invention solves this problem by providing a mechanism to run a personal certificate server that can support different levels of security and is under control of the owner of the certificates rather than control of the Web browser. The certificates can be made available to many browsers as well as applications and remote operations over the network. Applications may perform many operations with the many different types of certificates a user may own.

Figure 4 illustrates of the current technology's approach to maintaining cryptographic keys using multiple key stores. In this example, a user begins with home system 400. Using Web browser 410 the user starts email application 420. Web browser 410 uses the PKCS#11 cryptographic method to create and manage cryptographic data. From email application 420, the user wishes to send an email message signed with a digital signature. Email application 420 makes signature request 450 of Web browser 410 to sign email message 470. The message is signed using PKCS#11 interface 430 and cryptographic data stored in key store 440. The signed email message is returned to email program 420, which then sends the message over dialup network connection 480 to Web server 490 for further processing.

Continuing the scenario of Figure 4, the same user now moves to office system 405. Office system 405 is configured to run email application 425 – a different program than the one on home system 400. Email application 425 makes signature request 455 of CSP interface 435, and received back signed message 475 that was encrypted with cryptographic data stored in key store 445.

The data in key store 445 and the data in key store 440 are different for two reasons. First, the cryptographic schemes used by interfaces 430 and 435 are different, and therefore require different key stores. Second, web browser 410 and web browser 415 each control the email program interface to the cryptographic system. Meanwhile, the user's cryptographic data is vulnerable to attack in two different locations.

In contrast, Figure 5 shows a method of one embodiment of the claimed invention that eliminates the problems of the traditional approach to cryptographic key management illustrated in Figure 4. In the method of the claimed invention, the user's cryptographic data is stored on Internet Aware Device (IAD) 550. In one embodiment of the invention IAD 550 is a Smart Card; for example a Java Card™. The user may use either home system 500 or office system 505 to initiate a secure email message. Email application 510 or 515, which use different cryptographic schemes as illustrated above, initiates a request for a digital signature in the form of URL request 520. Even though both applications use different cryptographic schemes they make the same URL call, passing different parameters to indicate the cryptographic mechanism to use to encode the data. Web server 530 receives each of the requests, and uses UPI interface 560 to perform the requested encryption, using data stored in data vault 570. Signed message 580 is returned as a response to the request by email applications 510 and 515. Next, email applications 510 and 515 each send signed message 690 back to Web server 530 for delivery. In this example of the claimed invention, the user has only one location for cryptographic data, data vault 570. Data vault 570 is accessible from any network because it can respond to a URL via the UPI of the claimed invention.

Figure 6 shows, in one embodiment of the invention, the steps to create a cryptographic key pair and sign a data item using the claimed invention. At step 600, the application calls to retrieve a list of available tokens. A data vault currently accessible on a network is called a token. In one embodiment of the invention at step 610, if the desired token is not available, perhaps because a card has been removed from the card reader, the application or user must select another token. Next, at step 620, the application gets information about the selected token, such as its object attributes. To use the token the application must create a session, which establishes its connection to the token. This is done at step 630. Step 640 requires the application to provide the user's PIN or other identification required to login to the session for a particular user. In this scenario the user wishes to first create a key pair, then use that key pair to sign a message. The key pair will be stored on the token for future use. The application provides the required information at step 650 and the key pair is generated. At step 660, the application specifies the cryptographic mechanism, key id and data to be signed. Finally, at step 670 the signed data is returned. The user's private key is stored in the token's memory for future use.

Embodiment of Computer Execution Environment (Hardware)

The claimed invention is a computer system composed of both hardware and software. The following detailed description of a general computer hardware environment is an example of a system that may be implement the claimed invention. This example is illustrative and does not limit the types of computers or computer systems that may be used to implement the claimed invention.

An embodiment of the invention can be implemented as computer software in the form of computer readable code executed on a general-purpose

computer such as computer 700 illustrated in Figure 7. A keyboard 710 and mouse 711 are coupled to a bi-directional system bus 718. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to processor 713. Other suitable input devices may be used in addition to, or in place of, the mouse 711 and keyboard 710. I/O (input/output) unit 719 coupled to bi-directional system bus 718 represents such I/O elements as a printer, A/V (audio/video) I/O, etc.

Computer 700 includes a video memory 714, main memory 715 and mass storage 712, all coupled to bi-directional system bus 718 along with keyboard 710, mouse 711 and processor 713. The mass storage 712 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. Bus 718 may contain, for example, address lines for addressing video memory 714 or main memory 715. The system bus 718 also includes, for example, a data bus for transferring data between and among the components, such as processor 713, main memory 715, video memory 714 and mass storage 712. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

In one embodiment of the invention, the processor 713 is a microprocessor manufactured by Motorola, such as the 680X0 processor or a microprocessor manufactured by Intel, such as the 80X86, or Pentium processor, or a SPARC™ microprocessor from Sun Microsystems, Inc. However, any other suitable microprocessor or microcomputer may be utilized. Main memory 715 is comprised of dynamic random access memory (DRAM). Video memory 714 is a dual-ported video random access memory. One port of the video memory 714 is coupled to video amplifier 716. The video amplifier 716 is used to drive the cathode ray tube (CRT) raster monitor 717. Video amplifier 716 is well known in

the art and may be implemented by any suitable apparatus. This circuitry converts pixel data stored in video memory 714 to a raster signal suitable for use by monitor 717. Monitor 717 is a type of monitor suitable for displaying graphic images. Alternatively, the video memory could be used to drive a flat panel or liquid crystal display (LCD), or any other suitable data presentation device.

Computer 700 may also include a communication interface 720 coupled to bus 718. Communication interface 720 provides a two-way data communication coupling via a network link 721 to a local network 722. For example, if communication interface 720 is an integrated services digital network (ISDN) card or a modem, communication interface 720 provides a data communication connection to the corresponding type of telephone line, which comprises part of network link 721. If communication interface 720 is a local area network (LAN) card, communication interface 720 provides a data communication connection via network link 721 to a compatible LAN. Communication interface 720 could also be a cable modem or wireless interface. In any such implementation, communication interface 720 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 721 typically provides data communication through one or more networks to other data devices. For example, network link 721 may provide a connection through local network 722 to local server computer 723 or to data equipment operated by an Internet Service Provider (ISP) 724. ISP 724 in turn provides data communication services through the worldwide packet data communication network now commonly referred to as the "Internet" 725. Local network 722 and Internet 725 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks

and the signals on network link 721 and through communication interface 720, which carry the digital data to and from computer 700, are exemplary forms of carrier waves transporting the information.

Computer 700 can send messages and receive data, including program code, through the network(s), network link 721, and communication interface 720. In the Internet example, remote server computer 726 might transmit a requested code for an application program through Internet 725, ISP 724, local network 722 and communication interface 720.

The received code may be executed by processor 713 as it is received, and/or stored in mass storage 712, or other non-volatile storage for later execution.. In this manner, computer 700 may obtain application code in the form of a carrier wave. In accordance with an embodiment of the invention, an example of such a downloaded application is the apparatus for debugging a virtual machine described herein.

Application code may be embodied in any form of computer program product. A computer program product comprises a medium configured to store or transport computer readable code or data, or in which computer readable code or data may be embedded. Some examples of computer program products are CD-ROM disks, ROM cards, floppy disks, magnetic tapes, computer hard drives, servers on a network, and carrier waves.

The computer systems described above are for purposes of example only. An embodiment of the invention may be implemented in any type of computer system or programming or processing environment, including embedded

devices (e.g., Web phones, etc.), wireless devices, and "thin" client processing environments (e.g., network computers) that support a virtual machine.

Thus, a method and apparatus for managing cryptographic keys using Uniform Resource Locator (URL) programming interface (UPI) have been described in conjunction with one or more specific embodiments. The invention is defined by the claims and their full scope of equivalent.

CLAIMS

1. A method in a computer system comprising:
storing cryptographic data on a portable device;
accessing said cryptographic data through an addressable interface.

2. The method of claim 1 wherein said cryptographic data comprises a session key.

3. The method of claim 1 wherein said cryptographic data comprises a public key.

4. The method of claim 1 wherein said cryptographic data comprises a private key.

5. The method of claim 1 wherein said cryptographic data comprises at least one digital certificate.

6. The method of claim 1 wherein said addressable interface operates over an interconnection fabric.

7. The method of claim 1 wherein said storing further comprising:
providing a protocol stack to process requests for said cryptographic data;
wherein said portable device comprises an internet aware device.

8. The method of claim 7 further comprising:
retrieving said cryptographic data if valid user credentials are presented.

9. The method of claim 8 further comprising:

generating a digital signature based on said cryptographic data.

10. The method of claim 7 wherein said protocol stack comprises:

providing a secure communication session;

encrypting data in said secure communication session using at least one cryptographic scheme;

decrypting data in said secure communication session using said at least one cryptographic scheme;

generating cryptographic keys in said secure communication session using at least one cryptographic scheme.

11. An apparatus comprising:

an internet interface unit configured to utilize the HTTP protocol;

a web server;

an internet aware device comprising a processing unit and a memory;

wherein said internet aware device can access a data vault.

12. The apparatus of claim 11 wherein said data vault comprises at least one cryptographic key.

13. The apparatus of claim 11 wherein said data vault comprises at least one digital certificate.

14. A computer program product comprising:

computer readable program code configured to cause a computer to store cryptographic data on a portable device;

computer readable program code configured to cause a computer to access said cryptographic data through an addressable interface.

15. The computer program product of claim 14 wherein said cryptographic data comprises a session key.

16. The computer program product of claim 14 wherein said cryptographic data comprises a public key.

17. The computer program product of claim 14 wherein said cryptographic data comprises a private key.

18. The computer program product of claim 14 wherein said cryptographic data comprises at least one digital certificate.

19. The computer program product of claim 14 wherein said addressable interface operates over an interconnection fabric.

20. The computer program product of claim 14 further comprising:

computer readable program code configured to cause a computer to provide a protocol stack to process requests for said cryptographic data;

wherein said portable device comprises an internet aware device.

21. The computer program product of claim 20 further comprising:

computer readable program code configured to cause a computer to retrieve said cryptographic data if valid user credentials are presented.

22. The computer program product of claim 21 further comprising:

computer readable program code configured to cause a computer to generate a digital signature based on said cryptographic data.

23. The computer program product of claim 20 wherein the functions of said protocol stack comprise:

computer readable program code configured to cause a computer to provide a secure communication session;

computer readable program code configured to cause a computer to encrypt data in said secure communication session using said at least one cryptographic scheme;

computer readable program code configured to cause a computer to decrypt data in said secure communication session using at least one cryptographic scheme;

computer readable program code configured to cause a computer to generate cryptographic keys in said secure communication session using at least one cryptographic scheme.

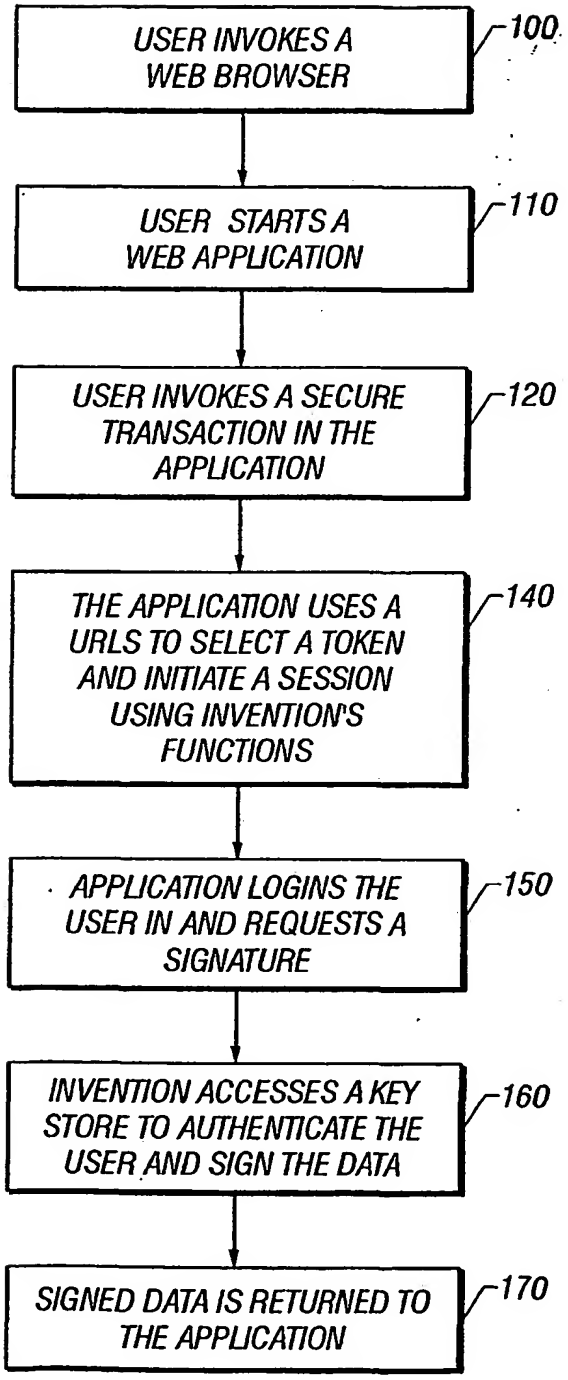

USER INVOKES A WEB BROWSER
100
USER STARTS A WEB APPLICATION
110
USER INVOKES A SECURE TRANSACTION IN THE APPLICATION
120
THE APPLICATION USES A URLS TO SELECT A TOKEN AND INITIATE A SESSION USING INVENTION'S FUNCTIONS
140
APPLICATION LOGINS THE USER IN AND REQUESTS A SIGNATURE
150
INVENTION ACCESSES A KEY STORE TO AUTHENTICATE THE USER AND SIGN THE DATA
160
SIGNED DATA IS RETURNED TO THE APPLICATION
170

FIGURE 1

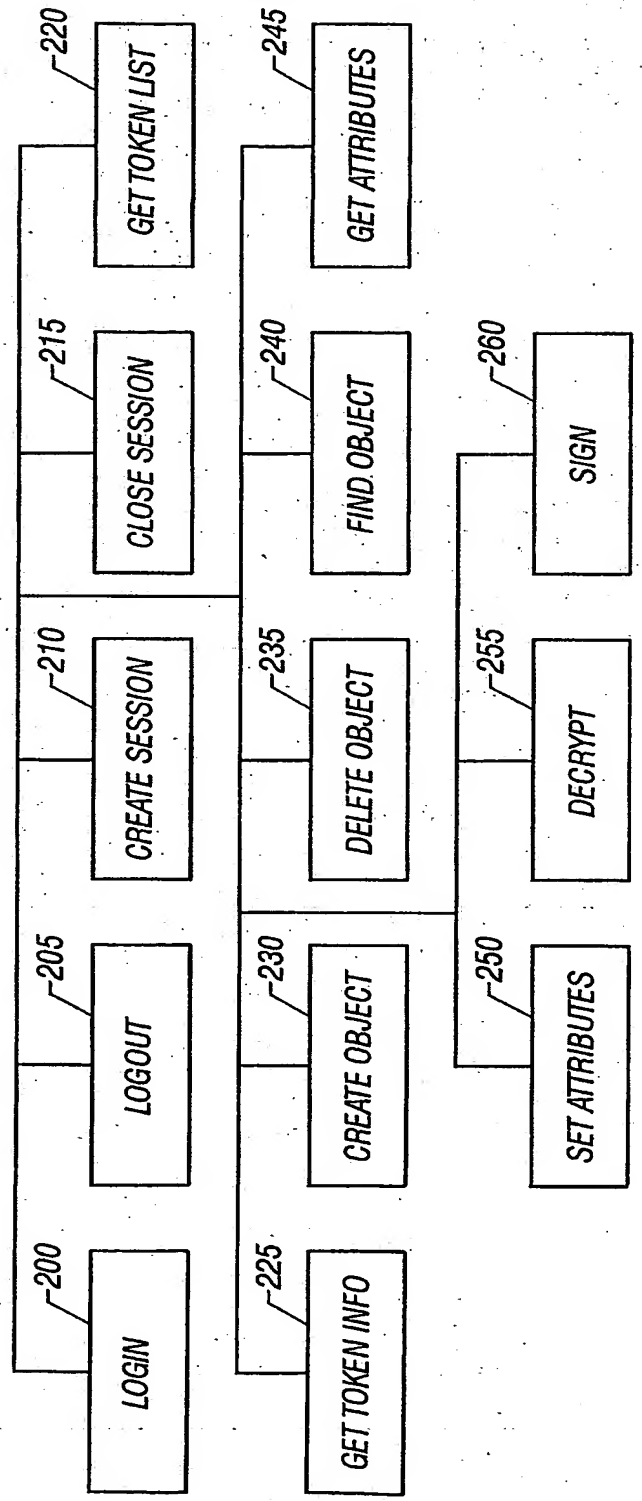

LOGIN
200
LOGOUT
205
CREATE SESSION
210
CLOSE SESSION
215
GET TOKEN LIST
220
GET TOKEN INFO
225
CREATE OBJECT
230
DELETE OBJECT
235
FIND OBJECT
240
GET ATTRIBUTES
245
SET ATTRIBUTES
250
DECRYPT
255
SIGN
260

FIGURE 2

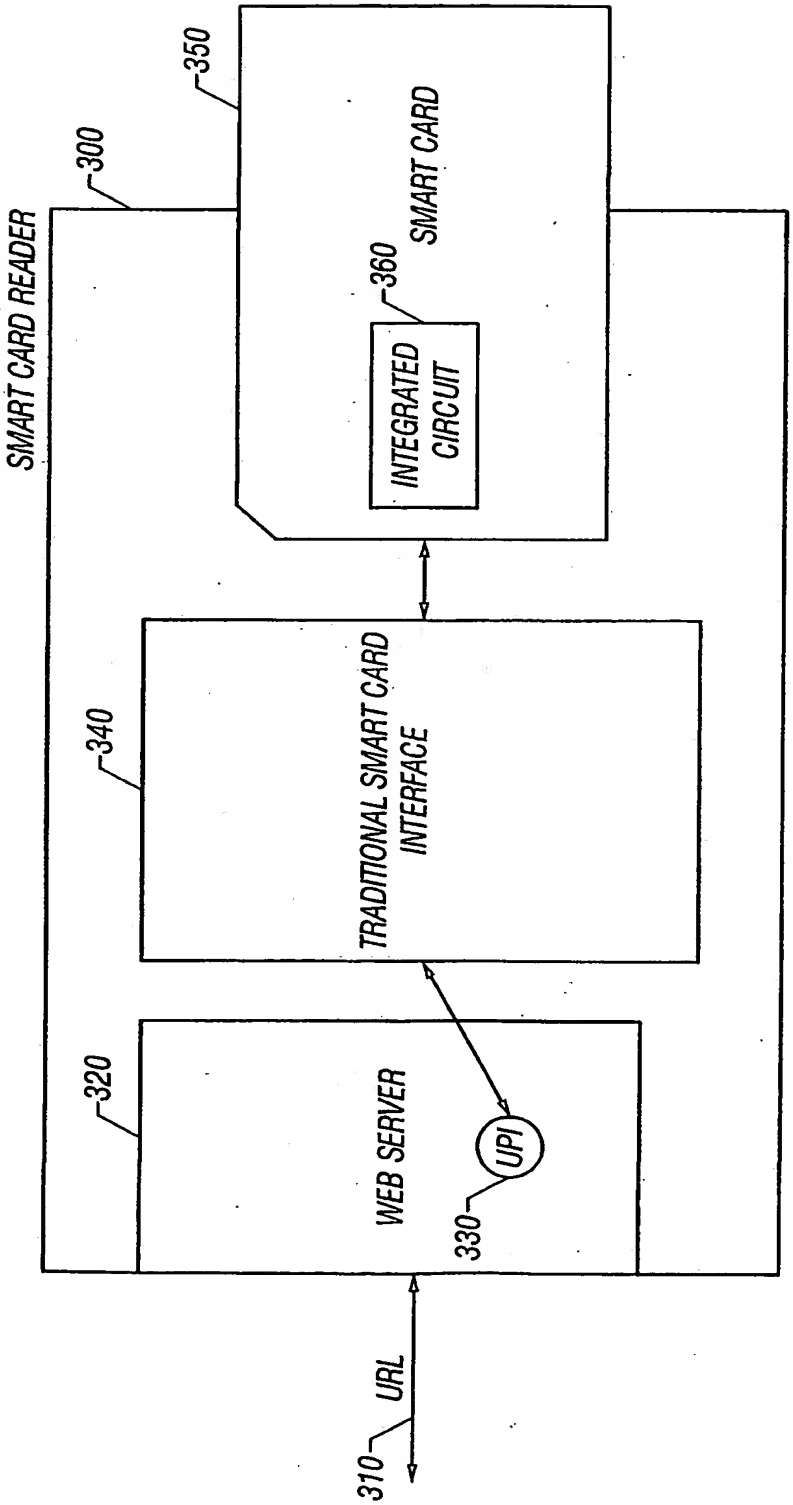

SMART CARD READER
300
350
SMART CARD
360
INTEGRATED CIRCUIT
340
TRADITIONAL SMART CARD INTERFACE
320
WEB SERVER
UPI
330
URL
310

FIGURE 3

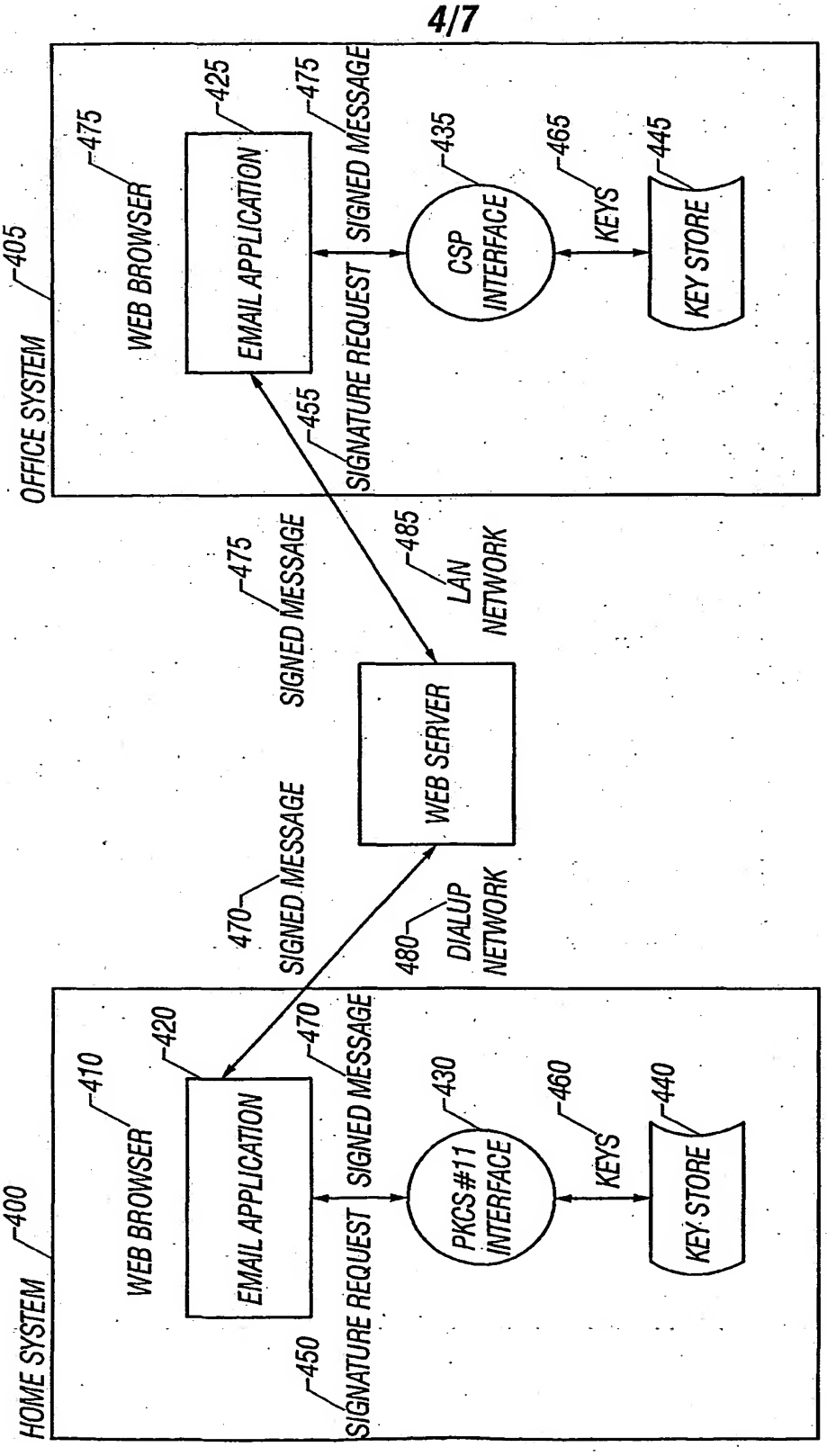

HOME SYSTEM
400
WEB BROWSER
410
EMAIL APPLICATION
420
SIGNATURE REQUEST
450
SIGNED MESSAGE
470
PKCS#11 INTERFACE
430
KEYS
460
KEY STORE
440
SIGNED MESSAGE
470
DIALUP NETWORK
480
WEB SERVER
SIGNED MESSAGE
475
LAN NETWORK
485
OFFICE SYSTEM
405
WEB BROWSER
475
EMAIL APPLICATION
425
SIGNATURE REQUEST
455
SIGNED MESSAGE
475
CSP INTERFACE
435
KEYS
465
KEY STORE
445

FIGURE 4

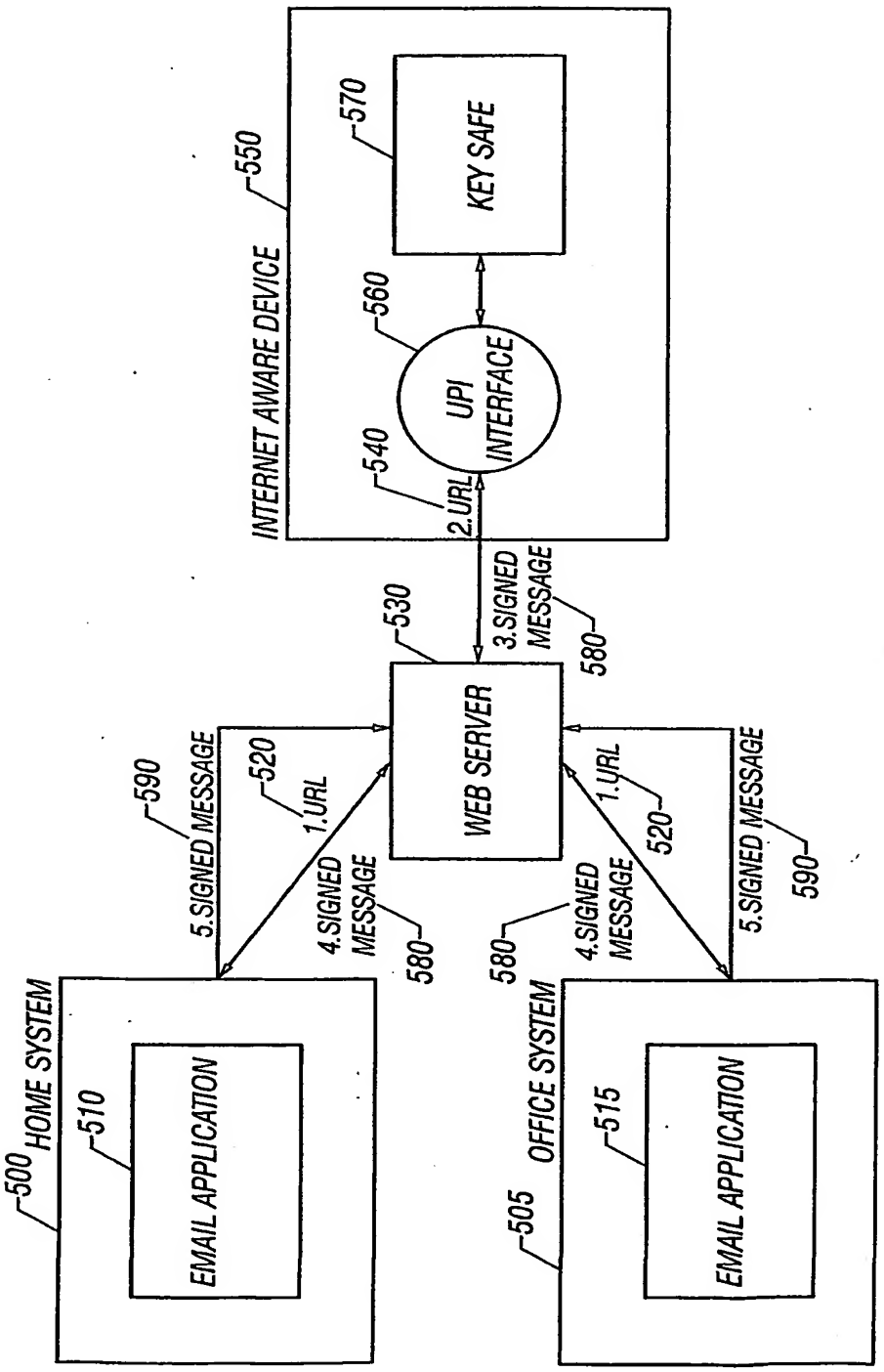

INTERNET AWARE DEVICE
550
KEY SAFE
570
UPI INTERFACE
560
540
2.URL
3.SIGNED MESSAGE
580
530
WEB SERVER
1.URL
520
5.SIGNED MESSAGE
590
4.SIGNED MESSAGE
580
HOME SYSTEM
500
EMAIL APPLICATION
510
OFFICE SYSTEM
505
EMAIL APPLICATION
515

FIGURE 5

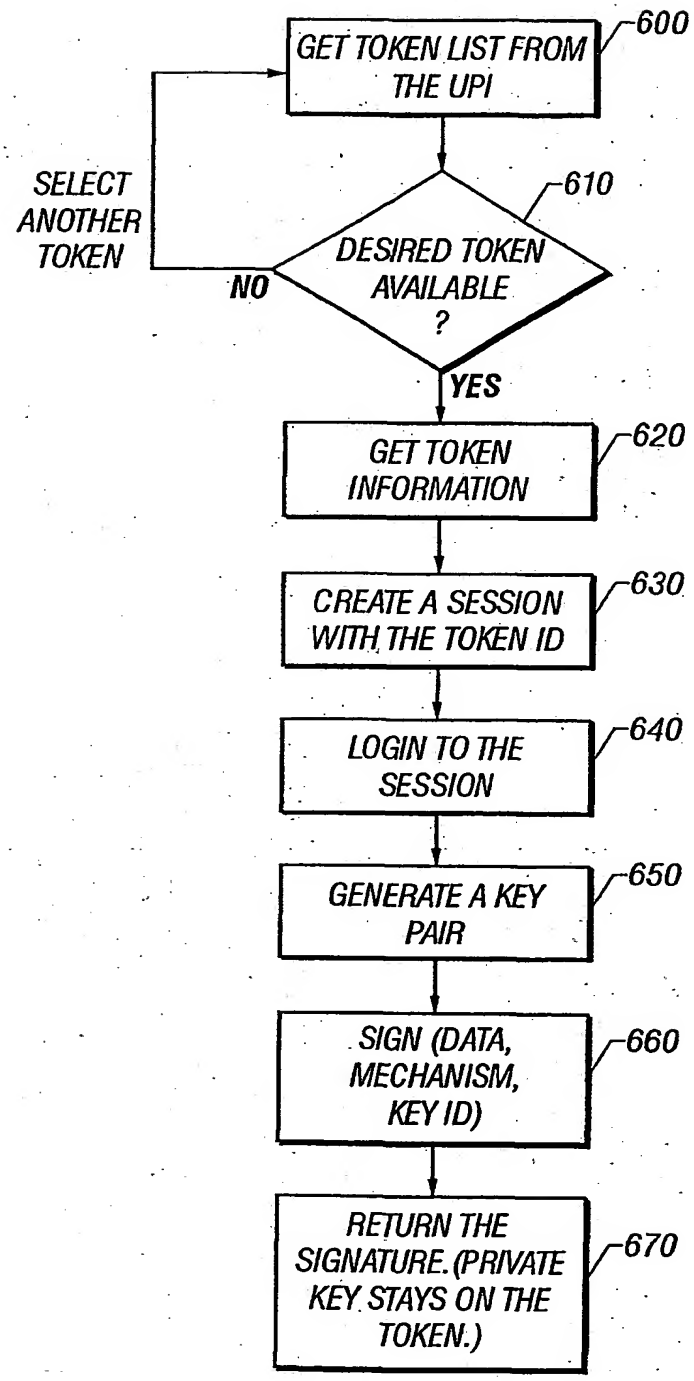

GET TOKEN LIST FROM THE UPI
600
SELECT ANOTHER TOKEN
610
DESIRED TOKEN AVAILABLE ?
NO
YES
GET TOKEN INFORMATION
620
CREATE A SESSION WITH THE TOKEN ID
630
LOGIN TO THE SESSION
640
GENERATE A KEY PAIR
650
SIGN (DATA, MECHANISM, KEY ID)
660
RETURN THE SIGNATURE. (PRIVATE KEY STAYS ON THE TOKEN.)
670

FIGURE 6

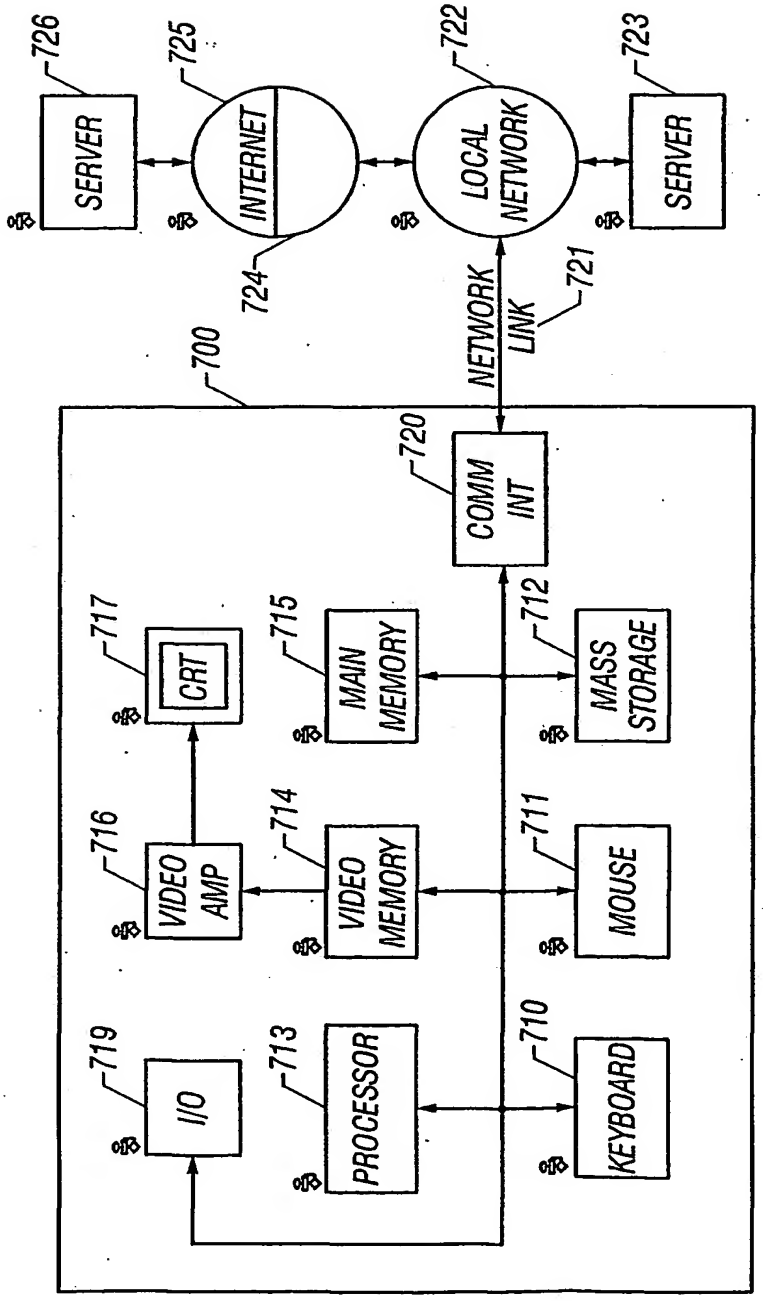

726
SERVER
725
INTERNET
722
LOCAL NETWORK
723
SERVER
724
NETWORK LINK
721
700
720
COMM INT
717
CRT
715
MAIN MEMORY
712
MASS STORAGE
716
VIDEO AMP
714
VIDEO MEMORY
711
MOUSE
719
I/O
713
PROCESSOR
710
KEYBOARD

FIGURE 7